Preamble

Please note the following very important details relating to input and output:

1) Read all input from the keyboard, i.e. use `stdin`, `System.in`, `cin` or equivalent. Input will be redirected from a file to form the input to your submission.

2) Write all output to the screen, i.e. use `stdout`, `System.out`, `cout` or equivalent. Do not write to `stderr`. Do NOT use, or even include, any module that allows direct manipulation of the screen, such as `conio`, `Crt` or anything similar. Output from your program is redirected to a file for later checking. Use of direct I/O means that such output is not redirected and hence cannot be checked. This could mean that a correct program is rejected!

3) Unless otherwise stated, all *integers* will fit into a standard 32-bit computer word. If more than one integer appears on a line, they will be separated by white space, i.e. spaces or tabs.

4) Unless otherwise stated, a *word* is a continuous sequence of lower case letters without any punctuation or other characters and, in particular, without intervening white space. As with integers, successive words will be separated by white space.

6) Unless otherwise stated, a *name* is a continuous sequence of letters without any punctuation or other characters and, in particular, without intervening white space. As with words, successive names will be separated by white space.

7) Unless otherwise stated, a *string* is a continuous sequence of characters without any intervening white space. As with words, successive strings will be separated by white space.

8) If it is stated that 'a line contains no more than *n* characters', this does not include the character(s) specifying the end of line.

9) All input files are terminated by a 'sentinel' line, followed by an end of file marker. This line should not be processed.

Program Names

These will be site specific.

W Letters 3 points

In this problem you are required to count the number of times a particular letter occurs in a piece of text. For example, how many times does the letter ‘g’ appear in ‘Programming Contest’? The answer, of course, is 2. How many times does the letter ‘n’ appear in ‘New Zealand’? Again the answer is 2 because we count ‘N’ and ‘n’ as being the same letter.

Input consists of a number of lines. Each line consists of a single lower case letter, a blank (space) and a sequence of between 1 and 250 characters. Input will be terminated by a line consisting of a single #.

Output will consist of the input letter followed by a single space, followed by a single number. The number will be a count of the occurrences of the letter, lower or upper case, in the text.

Sample input

```
g Programming Contest
n New Zealand
x This is quite a simple problem.
#
```

Sample output

```
g 2
n 2
x 0
```

X Transactions 3 points

Whenever somebody goes to an ATM to withdraw or deposit money, a calculation has to be done to keep the person's bank balance correct. Your task in this problem is to do such calculations. There is a bank rule that says that a customer may not have an overdraft of more than $200, so any withdrawal that would take the balance below –200 must be stopped. (A minus sign is used to indicate an overdraft, or negative balance).

Input consists of a number of lines, each representing a transaction. Each transaction consists of an integer representing the starting balance (between –200 and +10,000), the letter W or the letter D (Withdrawal or Deposit), followed by a second integer representing the amount to be withdrawn or deposited (between 5 and 400). Input will be terminated by a line containing 0 W 0.

Output consists of one line for each line of input showing the new balance for each valid transaction If a withdrawal would take the balance below -200, the output must be the words ‘Not allowed’.

Sample input

```
10000 W 10
-200 D 300
50 W 300
0 W 0
```

Sample output

```
9990
100
Not allowed
```

Y Dates 3 points

Dates are often expressed as a sequence of numbers to represent the day, month and year. The 14th August 2004, for example, can be expressed as 14 8 2004. Programs that accept dates as input have to check that the numbers supplied are valid, which is what this problem is about.

What could be wrong with a date? Well here are some obvious examples:
12 0 2004 There is no month 0.
32 1 1976 January has only 31 days.
29 2 1974 1974 was not a leap year so there was no 29th February.

Hopefully you know that most months have 31 days, except for April, June, September and November which have 30, and February which has 28 (29 in a leap year). Every 4th year is a leap year, so if you can divide the year number exactly by 4 then it is a leap year. Well, that is normally correct, but at the turn of a century, where the year can be divided exactly by 100, it is not a leap year. There is one more exception though! You may remember that the year 2000 was a leap year because it can be divided exactly by 400.

Input will consist of a number of dates, each on a separate line. Each date will consist of three non-negative integers, the first to represent the day, the second the month and the third the year. The day and month will always be less than 100, and the year will always be between 1700 and 2200, inclusive. Input will be terminated by a line containing 0 0 0.

Output will be one line for each line of input, the line containing just one word. If the numbers represent a valid date, the word ‘Valid’ will be output, otherwise the word ‘Invalid’ will be output.

Sample input

```
14 8 2004
12 0 2004
32 1 1976
29 2 1974
29 2 1976
0 0 0
```

Sample output

```
Valid
Invalid
Invalid
Invalid
Valid
```

Z Words 3 points

A nasty virus has infected my computer. Its effect has been to attack all my text files and reverse every word in them. Your job in this problem is to write the code to restore my text files to their original condition.

As far as the virus was concerned, a word was any sequence of characters that ended with a space or an end of line character. You will see what I mean when I tell you that the first line in one of my files was:

```
Get ready for the New Zealand Programming Contest.
```

The virus turned this into:

```
teG ydaer rof eht weN dnalaeZ gnimmargorP .tsetnoC
```

See how it included the full stop with 'Contest' and put it before the letters?

Input will consist of a number of lines, each containing up to 250 characters. Words will be separated by single spaces, i.e. not by tabs, double spaces or other characters. Words may be of any length. Input will be terminated by a line containing a single #.

Output will consist of one line for each line of input. In the output line, each word (as defined in paragraph 2 above) in the input line will be reversed.

Sample input

```
teG ydaer rof eht weN dnalaeZ gnimmargorP tsetnoC no .ht41
I like ice-cream.
#
```

Sample output

```
Get ready for the New Zealand Programming Contest on 14th.
I ekil .maerc-eci
```

A Word Ratios 10 points

Many 'IQ' tests have questions of the form: `king : queen :: president : ?`, where the 'correct' answer (in USA anyway) is '`first lady`', which says a lot about IQ tests and western culture. Because these tests are so culture laden, psychologists at the University of Northern Southwestland have devised a similar test, based on the positional difference between the letters in the words. Thus a typical problem might be: `cat : dog :: emu : ?` to which the answer is '`fah`' because to go from '`cat`' to '`dog`' you advance the first letter by 1, the second by 14, and the third by 13. So '`cat`' to '`dog`' = '`emu`' to '`fah`'. However, these same psychologists are somewhat arithmetically challenged, so they are never quite sure that they have got the right answer. This is where you come in.

Write a program that will read in triples of words such as the ones above and determine the fourth word according to the rules outlined. Consider that the lower case alphabet wraps around at both ends, i.e. '`a`' succeeds '`z`' and '`z`' precedes '`a`'.

Input will consist of a series of problems. Each problem will consist of three words on a line (see the definition of a *word* in the Preamble), separated by single spaces. All the words on a line will be the same length (not more than 20 letters), but words on different lines may be of different lengths. Input will be terminated by a line consisting of a single '#' character.

Output will consist of one line for each line of input consisting of the three words given in the input followed by the 'answer', all separated by single spaces. Note that the answer must also be a word, i.e. it must be of the same length as the three input words and consist only of lower case letters.

Sample input

```
cat dog emu
frog wolf bear
```

Sample output

```
cat dog emu fah
frog wolf bear sbxq
#
```

B Winter Festival 10 points

The winter solstice has long been an important festival in Northern Europe, in fact it had been celebrated for many thousands of years before the Christians took it over and rebranded it as Christmas. The first settlers in New Zealand were from tropical countries so the solstices were not an issue for them. Hence it was not until the arrival of the white settlers some time later that Christmas became important. However, as with many imports, something was lost in the translation — in this case the timing. Midsummer is a time of holidays and beach parties and no one wants to be reminded of the impending arrival of winter, particularly in the far south of the country.

Because of this, the residents of Oban on Stewart Island decided this year that they would have a midwinter party. (They like having parties and figure that any excuse is better than none.) In accordance with custom, everyone was asked to bring along a small gift. These were distributed pretty well at random, but someone had the bright idea of recording the donor and recipient of each parcel. Given the community spirit there were no pikers, i.e. everybody brought a present and everybody received one, so there must have been at least one complete cycle of giving (A gave to B; B gave to C; .. gave to A), however determining these cycles was a bit beyond them, at least in the immediate aftermath of the party, so this is where you come in.

Write a program that will read in details of gifts and determine all the cycles involved. You can assume that everybody both gives and receives exactly one present.

Input will be descriptions of several parties, each containing details of two or more gifts. Each gift appears as a pair of *names* on a line, separated by a single space, meaning that the first person mentioned gave something to the second person mentioned. A name will conform to the specification given in the preamble and will contain no more than 20 letters. You can assume that there will never be more than 200 people at a party. The list of gifts for a particular party will be terminated by a line consisting of a single '#' character. The sequence of parties will also be terminated by such a line.

Output will consist of several lines for each party — a heading line followed by a succession of 'cycle' lines, as many as necessary. The heading line consists of the words 'Party number', followed by a space and the party number, a running number starting at 1. A cycle line consists of a series of names interspersed with the word 'to', all separated by spaces, and terminated by a period ('**.**'). The first and last names in any cycle must be the same, and the first names in the list of cycles must appear in the same order as in the input. Separate successive parties by a blank line. See the example below.

Sample input

```
Marie Sally
Fred Alice
Bob Gordon
George Marie
Gordon Fred
Sally George
Alice Bob
#
Alice Bob
Marie Sally
Fred Alice
Bob Gordon
George Marie
Gordon Fred
Sally George
#
#
```

Sample output

```
Party number 1
Marie to Sally to George to Marie.
Fred to Alice to Bob to Gordon to Fred.

Party number 2
Alice to Bob to Gordon to Fred to Alice.
Marie to Sally to George to Marie.
```

C Scrabble Scoring 10 points

Scrabble™ is a word game which has now sold in excess of 100 million sets in 121 countries and 29 languages. The game is played by placing tiles, each of which is inscribed with a letter, on a board, according to some simple rules, which we will not bother about now. The values of the tiles in the English version are:

10: Q, Z 8: J, X 5: K 4: F, H, V, W, Y 3: B, C, M, P 2: D, G
1: A, E, I, L, N, O, R, S, T, U

The board consists of a 15×15 grid of squares. Some of these squares are coloured and there is a bonus for using them. Letter bonus squares (2L, 3L) multiply the value of the letter placed on them by two or three respectively; word bonus squares (2W, 3W) multiply the score of the entire word (after any relevant letter multipliers) by two or three respectively. Thus if I had placed the word 'BANQUET' as shown on the right of the figure below then I would score 84 (3+1+1+10*2+1+1+1)*3. If I had played 'BANQUETS' starting in the same place I would have scored 261 (3+1+1+2*10+1+1+1+1)*3*3.

Bonus squares are shown below for the top left quadrant of the board and are symmetrically placed on the rest of the board, i.e. the board is reflected about column H and row 8.

	A	B	C	D	E	F	G	H
1	3W			2L				3W
2		2W				3L		
3			2W				2L	
4	2L			2W				2L
5					2W			
6		3L				3L		
7			2L				2L	
8	3W			2L				2W

	A	B	C	D	E	F	G	H
1	**B**	**A**	**N**	**Q**	**U**	**E**	**T**	3W
2		2W				3L		
3			2W				2L	
4	2L			2W				2L
5					2W			
6		3L				3L		
7			2L				2L	
8	3W			2L				2W

A play is denoted by specifying a starting position and orientation (row, column for horizontal words and column, row for vertical words) and the word. In actual play one would also need to worry about tiles already on the board, blank tiles, tiles in adjacent squares, bonus points for playing all the letters on your rack and so on, but we will ignore those details for this problem.

Input will consist of a series of lines, each denoting a play. Each line will start with the designation of the starting position of the word followed by a space and the word itself — a sequence of 2 to 15 upper case letters. The placement of the word will be such that it will fit on the board. Rows will be designated by a number in the range 1 to 15, columns by an upper case letter in the range 'A' to 'O'. If the row is specified first then the word is played horizontally, if the column is specified first then the word is played vertically. The sequence of plays will be terminated by a line containing a single '#'.

Output will consist of one line for each play in the input, consisting of the play itself, followed by a space and the score for that play, as outlined above.

Sample input

```
15H BANQUET
O1 BANQUETS
#
```

Sample output

```
15H BANQUET 84
O1 BANQUETS 261
```

D Holiday Costing 10 points

Resort hotels often offer "stay/pay" deals where after so many nights you get some nights free. For example, in a stay 8 / pay 7 deal, if you stay 8 nights then you only pay for 7 (one is free). If you stay 9 then you pay for 8. Often these deals can accumulate. For example, a stay 6 pay 4 deal that can be repeated three times means that stay 12 pay 8 and stay 18 pay 12 deals also exist. Some hotels offer multiple stay pay deals, however these cannot be combined.

Write a program that will calculate the best rate for a given stay, i.e. that will choose the best (possibly repeated) deal for a given length of stay.

Input consists of a number of sets of data, each relating to one hotel. The first line for each hotel is the name of the hotel, a sequence of up to 20 characters, possibly including spaces. This is followed by between 0 and 10 lines of stay/pay deals. Each stay/pay deal consists of three *integers* in the range 1 to 99: stay, pay, number of repeats. The stay/pay section is terminated by a line consisting of three zeroes (0 0 0). Following the stay/pay section is a series of integers, one per line, indicating some number of nights (also in the range 1 to 99), terminated by a line containing a single 0. This line should not be processed. The sequence of hotels is terminated by a line consisting of a single #.

Output consists of one line for each number of nights in the input, indicating the minimum number of nights that have to be paid for. Note that this minimum must be made up of a combination of standard deals (pay for as many nights as you stay) and at most one stay/pay deal, repeated if necessary. Match the sample output format given below.

Sample input

```
Hotel xyz
8 7 3
10 8 1
0 0 0
1
12
23
24
0
Hotel abc
0 0 0
11
0
#
```

Sample output

```
Stay 1 night at Hotel xyz, pay 1.
Stay 12 nights at Hotel xyz, pay 10.
Stay 23 nights at Hotel xyz, pay 21.
Stay 24 nights at Hotel xyz, pay 21.
Stay 11 nights at Hotel abc, pay 11.
```

F Stuttering Strings 30 points

The residents of a remote Himalayan monastery believe that strings of repeated syllables in their language have mystic power, particularly if they can produce chants containing all possible strings of a given length. Because outsiders are not even allowed to know the syllables, they have replaced them with single characters such that the resulting strings can be converted into chants easily. They have developed the following rules.

1) there are only two characters '*' and '!'.
2) In any given situation there is a fixed length of word.
3) To impose some structure on the sequences there can never be more than some number of each character in succession.
4) For completeness, we need all the possible sequences of characters that will match these rules.

Write a program that will read in the length of the sequence and the maximum number of adjacent '*'s and '!'s and produce all acceptable sequences. These should be in lexicographic sequence, where, for this problem, '*' is deemed to precede '!'.

Input consists of sequence of lines each containing 3 *integers*: n ($1 \le n \le 10$), *nstar* and *nbang*, and is terminated by a line containing 3 zeroes (0 0 0).

Output consists of, for each problem in the set, a header line starting with 'Sequence set ', followed by a running number starting at 1, followed by the ordered list of such sequences, i.e. those with no more than *nstar* consecutive '*'s and no more than *nbang* consecutive '!'s, one per line, followed by a blank line.

Sample input

```
1 1 1
3 0 0
2 10 1
0 0 0
```

Sample output

```
Sequence set 1
*
!

Sequence set 2

Sequence set 3
**
*!
!*
```

G Dart Scoring 30 points

The game of darts involves throwing darts at a darts board. A darts board is divided into 20 sector, numbered from 1 to 20. Each segment contains single, double and triple scoring areas, so, for example, in segment number 5 it is possible to score 5, 10 or 15 points depending on the area which the dart lands in. In the centre of the board are two concentric circles: the bull's eye (worth 50 points) and the outer bull (worth 25).

The game of 301 is played by two players, who take it in turns to throw groups of three darts. For each player, scoring does not start until they score a double (the bull's eye counts as a double). At the end of each group of three darts the total of those darts is deducted from that player's score (which counts down from 301).

To finish the game, a player must make their score exactly 0 by scoring a double. Hence a turn finishes, and no more darts are thrown, once the score drops below 2. If the final score is 0 and the last dart was a double, then the game is over and that player is the winner, otherwise the game continues and the player's score reverts to what it was at the beginning of the turn. Thus, if your score at the beginning of your turn is 10, then you should attempt to throw a double 5 and end the game. If you throw triple 3, single 9, single 10, or more than 10 then your turn ends and your score remains at 10. If you throw anything else then that throw is temporarily subtracted from your score and you continue attempting to end the game on a double. If you do then you win, if you don't and your score drops below 2, then your turn ends and your score remains as it was at the beginning of your turn (in this case 10).

Write a program to score darts games.

Input consists of details for a series of games, with each game consisting of a series of throws occupying one or more lines. Each line (except possibly the last one in a game) consists of 10 throws where each throw consists of a specifier and a comma. A throw specifier is one of the following:

- an S, D or T followed by number in the range 1 to 20 indicating a single, double or triple area in that sector.
- a B (bull's eye) or OB (outer bull).
- an M if the dart missed (scores 0).
- a # to indicate the end of a game.

The sequence of games is terminated by a line consisting of a single '#'.

Output consists of several lines for each game. The first line contains the word 'Game', followed by a space and the game number — a running number starting at 1. This is followed by as many lines as necessary, with each line showing the scores at the end of each round. (Assume that the game starts with an empty zeroth round, so the first line is always ' 301 301'. Each line shows the scores for the two teams right justified in a field of width 5. The last line will contain the score of the losing team and the word 'Wins' right justified in a field of width 5 under the appropriate column. Separate successive games by a blank line.

Sample input

```
D20,M,T20,T20,S19,B,T1,S5,T5,T20,
T20,T20,T20,T20,T20,S1,S20,B,#,
D20,M,T20,T20,S19,B,T7,S5,T5,T20,
T20,T20,T20,T20,D20,#,
#
```

Sample output

```
Game 1
  301  301
  201  251
  178   71
  178 Wins

Game 2
  301  301
  201  251
  160   71
 Wins   71
```

H A Voting Protocol 30 points

Consider the following system for elections in which multiple candidates are to be elected. Suppose that there are k positions to fill. Each voter provides a ranked list of their k most preferred candidates, identified by a single lower case letter, with their first choice in position 1, their second choice in position 2, and so on.

The winning candidates are then chosen in a series of rounds. In each round, each voter votes for the first candidate in their list of preferences who has not yet been selected. The candidate with the greatest number of votes is added to the list of selected candidates. In case of a tie, all the tied candidates are added to the set of selected candidates, unless this would result in more than k selections. In this case, the tie is broken on alphabetical order. That is, if two further candidates are required, and 'a', 'x', and 'z' all have equally many votes, then 'a' and 'x' are selected. The election ends when k candidates have been selected.

Input will consist of a number of elections. The first line of data for an election will contain two *integers* — the number of positions to be filled, k ($1 \leq k \leq 26$), and the number of voters, n ($1 \leq n \leq 1000$). This will be followed by n lines each containing a sequence of k lower case letters, without repetitions, i.e. noone can vote for the same person twice. The file will be terminated by a line containing two zeroes (0).

Output will consist of a series of lines, one for each election. Each line will consist of the word ‘Election’, a space, the number of the election (a running number starting at 1), a colon and a space (‘: ’) and the list of selected candidates in selection order (m before g in Election 1), and alphabetically for selections in the same round (as in Election 2).

Sample input

```
2 3
gm
mg
mv
4 4
kqxj
qxjk
xjkq
jkqx
0 0
```

Sample output

```
Election 1: mg
Election 2: jkqx
```

I Family Trees 30 points

Nowadays New Zealanders come from a variety of backgrounds. This means that any given person's ancestry could include significant contributions from Maori, the Pacific Islands, various western European nations and quite possibly some south east Asian as well. Given a suitable database of family lineages, it should be reasonably simple to determine the proportions of these contributions for any given individual. However, the world has never been simple, so really all we can determine, for any given pair individuals, is how much the earlier one contributed to the later one (if any). Your task is to write a program to do this.

Input will consist of a series of family trees and queries over that family tree. A family tree consists of a series of lines terminated by a line consisting of a single '#'. Each line contains three different *names* (each containing fewer than 20 letters) representing the person and his or her parents. Any name will appear first at most once (i.e. everyone has exactly zero or two listed parents) and there are no cycles (i.e. no-one is their own ancestor). The family tree is followed by a series of queries terminated by a line consisting of a single '#'. Each query is a pair of names that may or may not have appeared in the family tree. The file is terminated by a line containing a single #

Output consists of one line for each query in the input. If either of the people named in the query is a direct descendent of the other then print the name of the descendent, a space, the word "is", another space, the fraction of ancestry (in simplest terms, as shown below), another space and then the name of the ancestor followed by a full stop ('.'). If this is not the case (or if either or both do not appear in the family tree) then the line consists of: "<name1> and <name2> are not related.". Leave a blank line between successive family trees.

Sample input

```
PebblesFlintstone WilmaFlintstone FredFlintstone
LisaRubble PebblesFlintstone BambamRubble
Don Jane FredFlintstone
Paul LisaRubble Don
#
Paul FredFlintstone
FredFlintstone BambamRubble
Jane Don
#
PebblesFlintstone WilmaFlintstone FredFlintstone
LisaRubble PebblesFlintstone BambamRubble
Don Jane FredFlintstone
Paul LisaRubble Don
#
Paul Jane
Jane Tarzan
#
#
```

Sample output

```
Paul is 3/8 FredFlintstone.
FredFlinstone and BambamRubble are not related.
Don is 1/2 Jane.

Paul is 1/4 Jane.
Jane and Tarzan are not related.
```

K The Jury is out 100 points

In most Western judicial systems that allow for trial by jury, the jury is selected from a panel of citizens chosen randomly from the electoral roll. However, both the prosecutor and the counsel for the defence have the right to veto any potential jury member that they think could be inimical to their case. This usually means that the resulting jury is fairly bland — typically white, male, Protestant, middle class and boring. It also means that potential jury members who have strong views, but who could possibly contribute meaningfully to the discussion, are rejected, precisely for that reason.

A better way to select a jury would be to determine in some way, preferably reasonably scientifically, each jury member's value to both the prosecution and defence. We can then define the *balance* of a person to be the difference between these two quantities and their *value* to be their sum. With these attributes we can attempt to choose a jury that is as balanced as possible, i.e. one for which the sum of the balances is as small as possible. If more than one selection is as balanced, then we will choose the one that contributes the most value, i.e. the sum of all the values is as large as possible. For example, consider choosing a jury of size 2 from the following panel of 4 — (20, 1), (1, 20), (10, 9), (10, 9). If we choose the first two we have a total balance of 0, whereas choosing the last 2 would give us a total balance of 2. If the last person was (9, 10), then choosing the last 2 would also give us a total balance of 0, but the value would be less (38 against 42), so we would still choose the first 2.

Write a program that will determine the best (i.e. most balanced) jury of a given size from a panel of potential members. If there is more than one such jury, then choose the one with the greatest total value. If there are still more than one then report any of them.

Input will be a series of jury panels. Each panel will start with an integer *k,* ($5 \le k \le 20$) on a line by itself, where *k* is the size of the jury to be chosen. This will be followed by no more than 100 lines, each containing two integers (in the range 1 through 20 inclusive) representing the value of that potential juror to the prosecution and defence respectively. These lines are implicitly numbered from 1 upwards and these numbers serve to identify the person. The list will be terminated by a line containing two zeroes (0, 0). The file will be terminated by a line containing a single zero (0).

Output will be series of pairs of lines, one pair for each jury panel. The first line in each pair identifies the jury and specifies the jury number (a running number starting at 1), the balance achieved and the value. Follow the example shown in the sample output. The next line contains, in order, a list of *k* integers representing the selected jury, separated by single spaces. Separate successive juries by a blank line.

Sample Input

```
5
 5  4
13 16
17 12
 6 18
 5 12
18  4
10 13
13  3
 1 13
0 0
0
```

Sample Output

```
Jury 1: balance 1, value 127
2 3 4 6 7
```

L Aibohphobia 100 points

This problem has been withdrawn for technical reasons.

M Box Toppling 100 points

A large manufacturing company packs its goods into boxes which are then stored in a square warehouse awaiting delivery. The boxes are all 1m × 1m × zm for some integer z ($1 < z < 30$). Initially all boxes stand on one end (i.e. with their long sides vertical), aligned with the walls of the warehouse. However the foreman wants to be able to see all of one face of every box for counting and identification. He cannot guarantee this while the boxes are standing on end, since a short box may be hidden behind one or more tall boxes, so he wants all the boxes to be laid flat. It is not obvious to the workers whether this can always be done, so your task is to determine this.

Note that toppling a box is equivalent to rotating it about an edge so that it effectively shifts it one metre from its original position, i.e. `..3.....` will produce `...111..`.

Input will consist of a series of scenarios. Each scenario starts with a line containing a single integer representing the length of the sides of the warehouse ($3 \le size \le 30$), followed by one or more lines, each containing three integers, specifying the locations (row column) and heights of boxes and is terminated by a line consisting of 3 zeroes (0 0 0) The sequence of scenarios is terminated by a line consisting of a single zero (0).

Output will consist of a series of lines, one for each scenario in the input. Each line will consist of the single word ‘Possible’ or ‘Impossible’ depending on whether it is or is not possible to topple all the boxes.

Sample input

```
4
1 1 3
1 4 3
4 1 3
4 4 3
0 0 0
4
1 1 3
1 4 3
4 1 2
4 4 3
0 0 0
0
```

Sample output

```
Impossible
Possible
```

N Superstitious Helicopter Pilots 100 points

Helicopter pilots are both conservative and superstitious. They have a set of preferred hops (e.g. 10 km East, and 20 km North) which are ranked according to their desirability. Further, there are some locations on which they are unwilling to land (e.g. the pig farm 20 km East and 50 km North of here).

Your task is to fill out a flight plan for a journey, if possible. This journey must only use the pilot's preferred hops, and must never land at a forbidden point. Furthermore, each hop must be chosen to be the most preferred one which allows the remainder of the journey to be completed.

So, for instance, if the goal is to travel 30 km East and 40 km North, using hops of, in order, (10, 10) and (0, 10) but not passing through the point (30, 30), then one could do 1 northerly hop followed by three north-easterly ones , i.e. (0, 10) 3*(10, 10); but a preferable flight plan is to do two north-easterly hops followed by an northerly hop and a final north-easterly hop, i.e. 2*(10,10) (0, 10) (10,10). On the other hand, if the point (20,30) were also forbidden then no flight plan would be possible.

In all cases, the journey to be undertaken consists of a fixed positive distance east, and a fixed positive distance north and each preferred hop consists of a non-negative distance east and a non-negative distance north, not both 0.

Input will consist of a number of journeys. Each journey will start with a line containing four integers: trip distance east, trip distance north, # of preferred hops, # of forbidden points. This will be followed by the preferred hops, in order, most preferred first, followed by the forbidden points, in both cases in the form: east-distance north-distance. Note that some hops may not be used and some forbidden points may be beyond the destination. The file will be terminated by a line containing 4 zeroes (0 0 0 0).

Output will consist of one line per journey. Each line will start with the word 'Trip', followed by a space, the journey number (a running number starting at 1), a colon and a space (': ') and a description of the trip in terms of hops. Note that successive identical hops are collected as in the above example. If the trip cannot be achieved then the description consists of the word 'impossible'.

Sample input

```
30 40 2 1
10 10
10 0
30 30
30 40 2 2
10 10
10 0
30 30
20 30
30 30 3 0
10 10
10 0
20 30
```

Sample output

```
Trip 1: 2*(10,10) (0,10) (10,10)
Trip 2: impossible
Trip 3: 3*(10,10)
```